

ERDC/ITL TR-03-6

Information Technology Laboratory



**US Army Corps
of Engineers®**
Engineer Research and
Development Center

Innovations for Navigation Projects Research Program

Analysis of the Added Mass of a Barge in Restricted Waters

Phase 2 Model

David R. B. Kraemer and Michael E. McCormick

August 2003

20031007 028

Analysis of the Added Mass of a Barge in Restricted Waters

Phase 2 Model

David R. B. Kraemer

*Department of Mechanical Engineering
University of Wisconsin-Platteville
1 University Plaza
Platteville, WI 53818-3099*

Michael E. McCormick

*Department of Civil Engineering
The Johns Hopkins University
3400 N. Charles Street
Baltimore, MD 21218-2686*

Final report

Approved for public release; distribution is unlimited

Prepared for U.S. Army Corps of Engineers
Washington, DC 20314-1000

Under INP Work Unit 33143

Monitored by Information Technology Laboratory
U.S. Army Engineer Research and Development Center
3909 Halls Ferry Road
Vicksburg, MS 39180-6199

ABSTRACT: A theoretical analysis of the added mass of barges in restricted waters is presented. The added-mass behaviors with varying waterway and barge parameters are predicted. A numerical technique is used to solve the potential-flow boundary-value problem. The analysis assumes that the Froude number is low. A restriction that the Froude number be less than or equal to 0.1 ensures that the surface waves will be small; this corresponds to a barge speed of 8 fps (2.4 m/s). For this low-speed condition, the added mass is unaffected by the barge speed. The separation between the barge and the wall is seen to have influence near the wall.

DISCLAIMER: The contents of this report are not to be used for advertising, publication, or promotional purposes. Citation of trade names does not constitute an official endorsement or approval of the use of such commercial products. All product names and trademarks cited are the property of their respective owners. The findings of this report are not to be construed as an official Department of the Army position unless so designated by other authorized documents.

DESTROY THIS REPORT WHEN IT IS NO LONGER NEEDED. DO NOT RETURN TO ORIGINATOR.

Contents

Preface	v
1—Introduction	1
Background.....	1
Hydrodynamics Analysis	1
Boundary value problem	3
Solution.....	4
Analysis of Kinetic Energy and Added Mass	6
2—Results	8
Effect of Barge Speed in Open Waters	8
Effect of Water Depth in Open Waters	8
Effect of Barge-Wall Separation and Barge Load	10
Rake Angle Effect.....	10
Comparison of the JOHB and SOHB	11
3—Discussion and Conclusions	12
References	15
Appendix A: FORTRAN Computer Code	A1
Appendix B: Sample Calculations	B1
SF 298	

List of Figures

Figure 1.	Model diagram—barge approaching quay wall at angle θ	2
Figure 2.	Reflections of physical barge to satisfy the bottom, water surface, and boundary conditions.....	5
Figure 3.	Sketch of barge surface discretized into panels.....	6
Figure 4.	Speed effect on added weight of loaded JOHB traveling in 18-ft-deep open waters with rake angle of 45 deg	9
Figure 5.	Water depth effect on added weight of loaded JOHB traveling parallel to wall at a speed of 3 fps with rake angle of 45 deg	9
Figure 6.	Wall-distance effect on added weight of loaded and unloaded JOHB traveling parallel to wall in 18-ft-deep water at a speed of 3 fps with rake angle of 45 deg.....	10
Figure 7.	Effect of bow rake angle on added weight of loaded JOHB operating in 18-ft-deep open waters at a speed of 3 fps.....	11

Preface

The work described in this report was authorized by Headquarters, U.S. Army Corps of Engineers (HQUSACE), as part of the Innovations for Navigation Projects (INP) Research Program. The study was conducted under Work Unit (WU) 33143, "Design of Innovative Lock Walls for Barge Impact." This research was initiated by Mr. Robert C. Patev, former Principal Investigator of WU 33143. Current Principal Investigator is Dr. Robert M. Ebeling of the U.S. Army Engineer Research and Development Center (ERDC) Information Technology Laboratory (ITL).

Dr. Tony C. Liu was the INP Coordinator at the Directorate of Research and Development, HQUSACE; Research Area Manager was Mr. Barry Holliday, HQUSACE; and Program Monitors were Mr. Mike Kidby and Ms. Anjana Chudgar, HQUSACE. Dr. Stanley C. Woodson, ERDC Geotechnical and Structures Laboratory (GSL), was the INP Program Manager.

This report was prepared by Dr. David R. B. Kraemer, Assistant Professor, Department of Mechanical Engineering, University of Wisconsin-Platteville, and Dr. Michael E. McCormick of the Department of Civil Engineering, The Johns Hopkins University, Baltimore, MD. Professors McCormick and Kraemer have backgrounds in ocean engineering and naval architecture.

The research was monitored by Dr. Ebeling, under the supervision of Dr. Charles R. Welch, Chief, Engineering and Informatic Systems Division; Dr. Jeffery P. Holland, Director, ITL; and Dr. David W. Pittman, Acting Director, GSL.

Commander and Executive Director of ERDC was COL James R. Rowan, EN. Director was Dr. James R. Houston.

1 Introduction

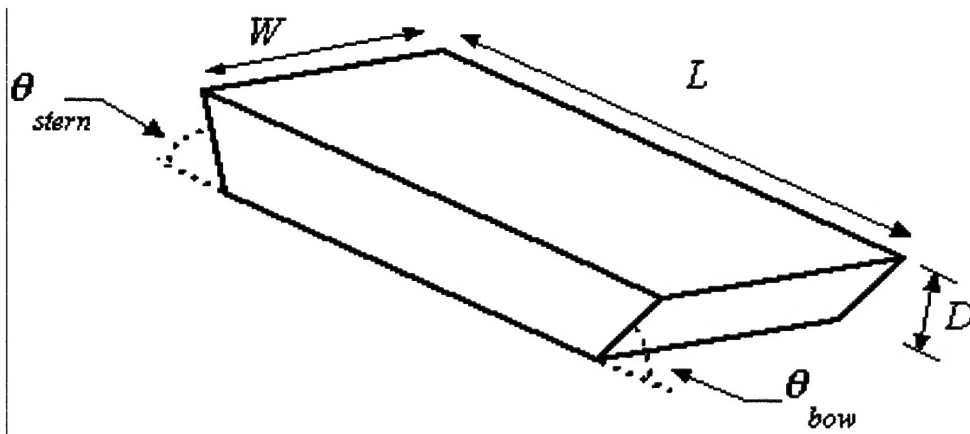
Background

In a simple towing-tank experiment, the free-surface excited by the forward motion of a barge model was studied (McCormick et al. 2002). Several rather important observations were made during this (approximately) two-dimensional (2-D) study of a rectangular barge in a narrow channel. First, when the towing speed was low enough, no dispersive surface waves were created. Instead, the free-surface elevation at the bow was found to increase as the model increased speed; at the stern, the free-surface drawdown increased with speed. This latter observation was somewhat obscured by wake effects. As the speed of the body increased, dispersive surface waves were created, producing rather interesting reactions on the model in the narrow channel.

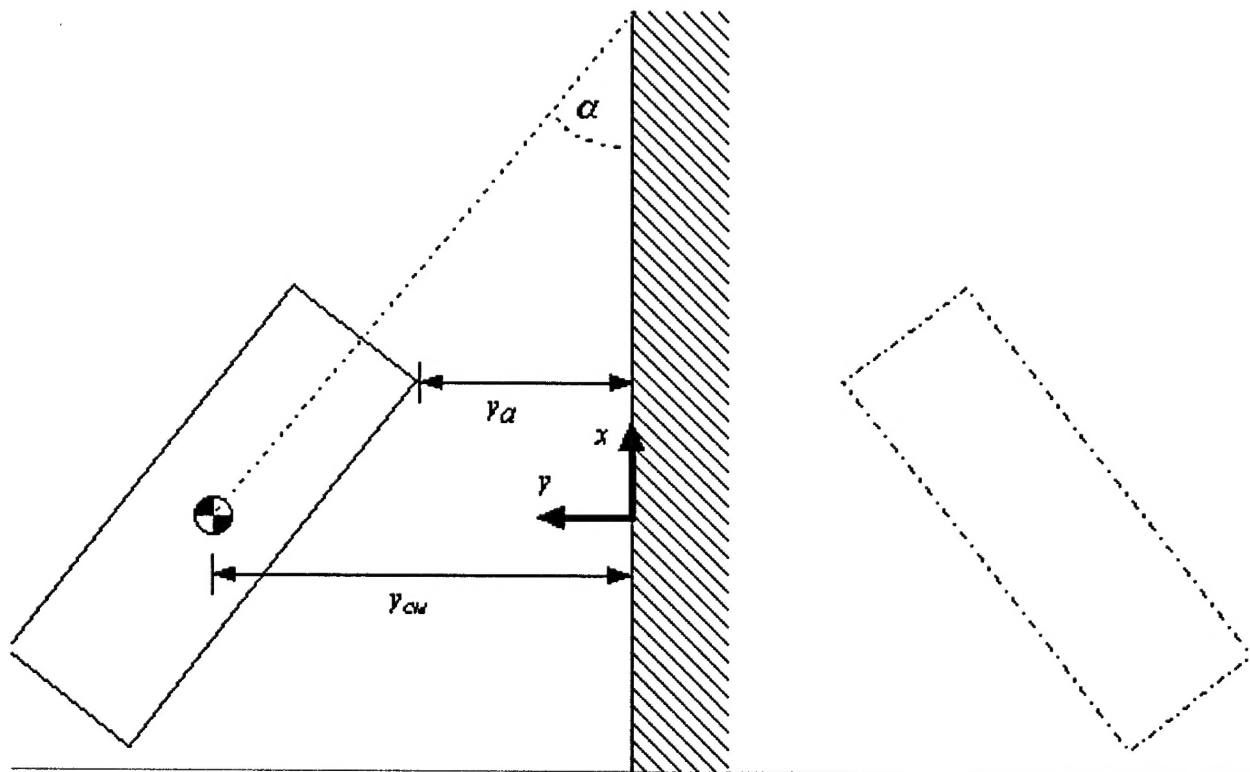
In Phase 1 of this study (McCormick et al. 2002), an analytical, 3-D added-mass model is presented. The analytical model is based on the low-speed free-surface behavior observed in the tank study. The Phase 2 analysis, described in this report, involves a numerical solution of the potential flow problem describing the flow around the barge in proximity to a wall.

Hydrodynamics Analysis

The mathematical model developed in this study describes a barge approaching a quay wall at angle α , as diagrammed in Figure 1. That figure presents an overhead water plane diagram of the approaching barge on the left-hand side and its mirror image on the right-hand side. The use of the mirror image allows for the effects of the presence of the quay wall. The inertial coordinate system has its origin at the wall at the mean water level, as shown in Figure 1. The y -axis is aligned through the center of gravity of the barge. The hydrodynamics are analyzed at the particular instant when the distance from the barge center of mass to the wall is equal to Y_{CM} . The z -axis (not shown) points vertically out of the water. Note that the barge is in motion in this study, while the coordinate system is fixed in space.



a. Barge geometry



b. Plan-view sketch of barge and wall geometry with mirror image

Figure 1. Model diagram—barge approaching quay wall at angle θ

For the general case considered here, it can be assumed that the barge has a constant acceleration (or deceleration), or that the barge is running at a constant speed. Hence, the barge speed is

$$\vec{V}(t) = \vec{V}_0 + \vec{a}_0 t \quad (1)$$

where

$$\vec{V}_0 = \text{barge's initial velocity}$$

$$\vec{a}_0 = \text{constant acceleration of the barge (negative for deceleration)}$$

Boundary value problem

The analysis uses a potential flow formulation of the problem. If the fluid is assumed to be incompressible (which is true for water in the conditions of interest to this problem) and if the flow is assumed to be irrotational (which is a valid assumption outside the boundary layer surrounding the body), the fluid velocity can be written as

$$\vec{U} = \nabla \phi \quad (2)$$

where

$$\vec{U} = \text{fluid velocity vector}$$

$$\nabla = \text{Laplacian operator}$$

$$\phi = \text{velocity potential}$$

The governing equation for a potential flow problem is Laplace's equation, which is

$$\nabla^2 \phi = 0 \quad (3)$$

The boundary conditions specify that there is no flow normal to a boundary surface. These boundary surfaces include the body surface, the wall surface, and the bottom surface. On the body surface, the normal component of the fluid velocity must equal the normal component of the body's velocity. This can be expressed mathematically as

$$\left. \frac{\partial \phi}{\partial n} \right|_s = \vec{V} \cdot \hat{n} \quad (4)$$

where

$$\begin{aligned} S &= \text{body surface} \\ \hat{n} &= \text{unit vector normal to the body surface} \\ \bar{V} &= \text{velocity of the body} \end{aligned}$$

Finally, the radiation boundary condition specifies that the velocity potential approaches zero far from the body, so that

$$\phi_{r \rightarrow \infty} = 0 \quad (5)$$

where r is the radial distance from the body.

The hydrodynamic analysis assumes that the barge is moving at low speed. This is quantified by the Froude number, which is

$$\text{Fr} = \sqrt{\frac{V}{gL}} \quad (6)$$

where V and L are the speed and the length of the barge, respectively, and g is the acceleration due to gravity.

For a small Froude number, which is valid for low speeds, the water surface acts like a rigid barrier. This is accomplished by reflecting the body surface about the water surface (see Figure 2). Doing so satisfies the boundary condition that there is no flow across the water surface. The wall and bottom boundary conditions are satisfied by similar reflections. A mirror image of the barge and its images imposes a no-flow condition across the wall. The bottom reflection ensures that there is no flow across the channel (or river) floor.

Solution

A Green's function is used to solve this boundary value problem. In this case, the Green's function is given by

$$G(\bar{x}, \bar{\xi}) = \frac{1}{r} \quad (7)$$

where r is the distance between the points \bar{x} and $\bar{\xi}$. The points \bar{x} and $\bar{\xi}$ can be anywhere in the fluid domain; typically, \bar{x} is a point in the fluid outside the body, while $\bar{\xi}$ is a point on the surface of the body. This function satisfies the governing equation and the boundary conditions. Physically, it is the velocity potential at point \bar{x} due to a unit source at point $\bar{\xi}$.

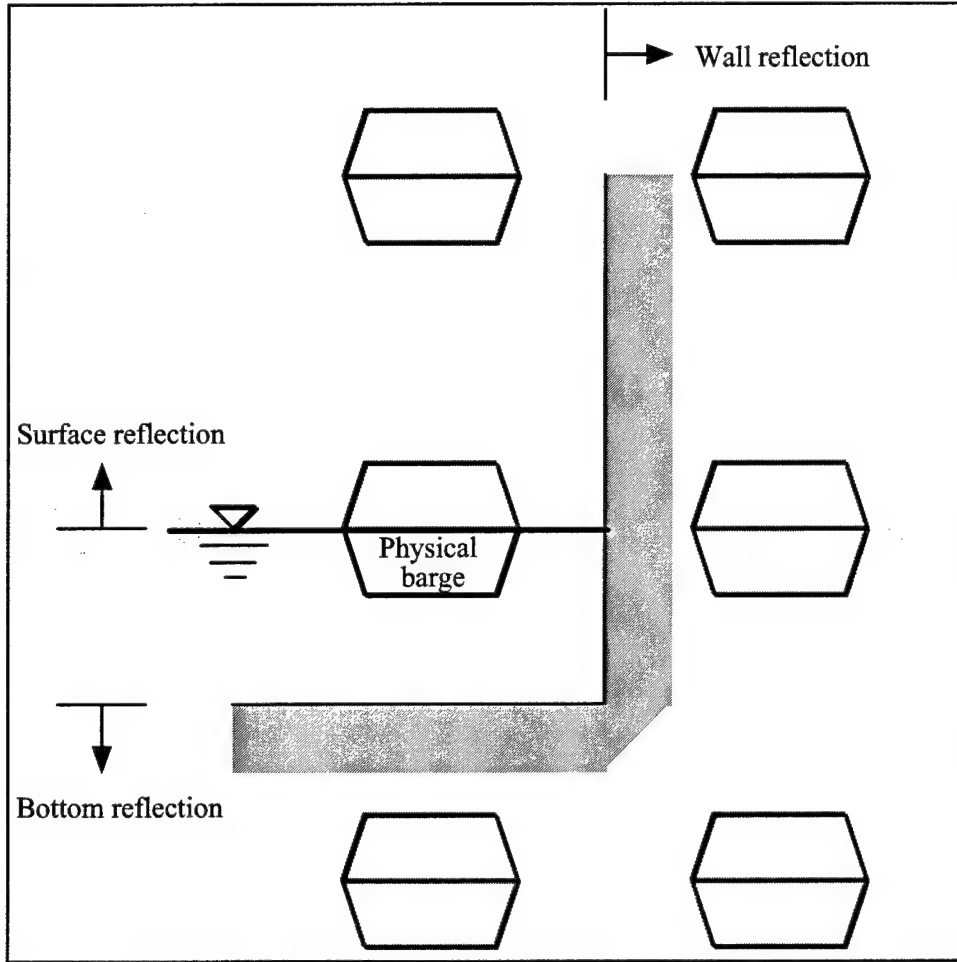


Figure 2. Reflections of physical barge to satisfy the bottom, water surface, and wall boundary conditions

The Green's function satisfies the governing differential equation (Equation 3) and the radiation boundary condition (Equation 5). The reflections (Figure 2) are used to satisfy the free-surface, bottom, and wall boundary conditions. The body boundary condition is satisfied using the method described below.

The velocity potential is represented as the potential due to a source distribution over the body surface S . Thus, the velocity potential at point \bar{x} is

$$\phi(\bar{x}) = \iint_S \sigma(\bar{\xi}) G(\bar{x}, \bar{\xi}) dS = \iint_S \sigma(\bar{\xi}) \frac{1}{r} dS \quad (8)$$

where σ is the source density distribution over the surface S . The normal velocity due to this source distribution is

$$\left. \frac{\partial \phi}{\partial n} \right|_s = -2\pi\sigma(\bar{x}) + \oint \frac{\partial}{\partial n} \left(\frac{1}{r(\bar{x}, \bar{\xi})} \right) \sigma(\bar{\xi}) dS \quad (9)$$

The $-2\pi\sigma$ term arises because the derivative in the integrand becomes singular when $\bar{\xi}$ approaches \bar{x} . This term is the value of the integral in the neighborhood of \bar{x} , and the integral is evaluated for the rest of the body surface. To satisfy the body boundary condition, this fluid velocity must satisfy Equation 4, so that

$$-2\pi\sigma(\bar{x}) + \oint \frac{\partial}{\partial n} \left(\frac{1}{r(\bar{x}, \bar{\xi})} \right) \sigma(\bar{\xi}) dS = \bar{V} \cdot \hat{n}(\bar{x}) \quad (10)$$

The body surface is discretized into quadrilateral panels in order to solve the problem for bodies of arbitrary geometry, as sketched in Figure 3. This allows the integral in Equation 9 to be approximated by a sum, and leads to a system of algebraic equations. These equations are solved numerically by the FORTRAN computer program presented in Appendix A (Section 1). The code currently uses 300 elements to approximate the surface of the physical barge. With the reflections due to the bottom, wall, and water surface, this increases the number of elements 12-fold. Since the reflected geometry is, by definition, symmetric about the wall and water-surface planes, one could revise the program code to take advantage of these symmetries, using only a quarter of the reflected elements. Since the computational resources required increase with the square of the number of elements, significant savings could be realized. However, for the simple barge geometries used in this study, the computational demands were modest enough to make this revision unnecessary.

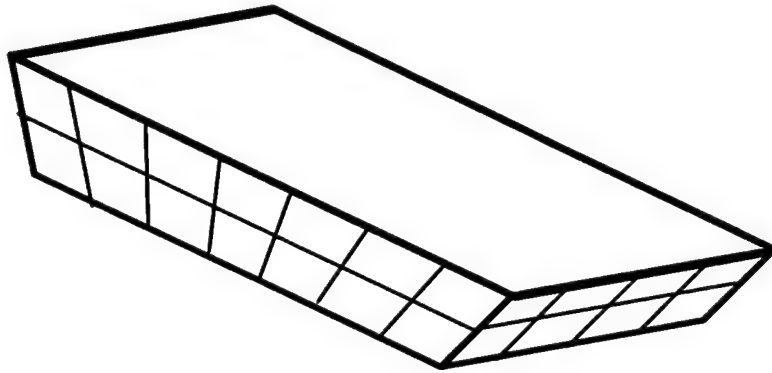


Figure 3. Sketch of barge surface discretized into panels

With the source distribution solved, one can then calculate the velocity potential and the fluid velocity at any point in the fluid domain. These expressions will be used to obtain the added mass excited by the physical barge.

Analysis of Kinetic Energy and Added Mass

As derived by Karamcheti (1966) and other books dealing with advanced fluid mechanics, the kinetic energy of a potential flow, T , that is caused by a body traveling at a speed V is mathematically expressed as

$$T = -\frac{\rho}{2} \iint_S \phi \frac{\partial \phi}{\partial n} dS = -\frac{1}{2} m_w V^2 \quad (11)$$

where ρ is the mass density of the ambient water and m_w is the added mass. The added mass, then, is

$$m_w = \frac{\rho}{V^2} \iint_S \phi \frac{\partial \phi}{\partial n} dS \quad (12)$$

Thus, with the velocity potential determined, the added mass can be calculated. Finally, this can be expressed as an “added weight” by multiplying by g , the acceleration due to gravity. Note that this quantity is traditionally calculated as a mass. However, for the present study, it is expressed as a weight, to facilitate comparison with measurements in units of force.

2 Results

Equation 12 is applied to the jumbo open-hopper barge (JOHB) in this report; calculations are performed using the FORTRAN programs listed in Appendix A. A comparison of one of the parametric conditions for the JOHB is made with the standard open-hopper barge (SOHB). Since the geometries of the two barges are similar, the behaviors with the parameters are also similar. For the two barges, the values listed in Table 1 apply.

Table 1
Comparison of Parametric Conditions, Jumbo and Standard Open-Hopper Barges

Parameter	JOHB		SOHB	
Length (L), ft [m]	195	[59]	175	[53]
Beam (W), ft [m]	35	[11]	26	[8]
Draft (D), ¹ ft [m]	Loaded – 10	3]	Max. – 10	[3]
	Unloaded – 3	[0.9]	Min. – 3	[0.9]
Displacement, short tons [M kg]	Loaded – 1,500	[1.4]	Loaded – 1,000	[0.9]
	Unloaded – 279	0.25]	Unloaded – 179	[0.16]
Rake angle (θ)	Variable for computations		Variable for computations	

¹ The loaded draft is used as the standard, assumed to be 10 ft; the unloaded draft is assumed to be 3 ft.

Effect of Barge Speed in Open Waters

Since the barge speed is assumed to have low values in confined waterways, the maximum speed value used herein is 4 fps (1.2 m/s). The added weight ($m_w g$) is presented in Figure 4 as a function of barge speed for the loaded JOHB traveling in open waters, where the operating water depth (h) is 18 ft. The JOHB parametric values are in Table 1. The added weight is chosen so that a direct comparison can be made with the displacement tonnage of the barge.

Effect of Water Depth in Open Waters

In Figure 5, the added weight is shown as a function of water depth at a position well away from the wall. Again, the parameters used are those of the loaded JOHB in Table 1.

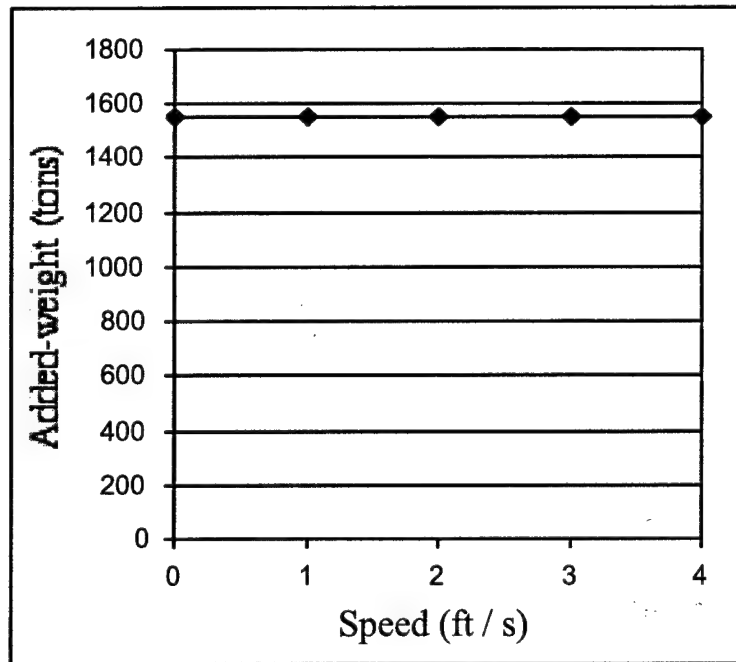


Figure 4. Speed effect on the added weight of the loaded JOHB traveling in 18-ft-deep open waters with a rake angle of 45 deg

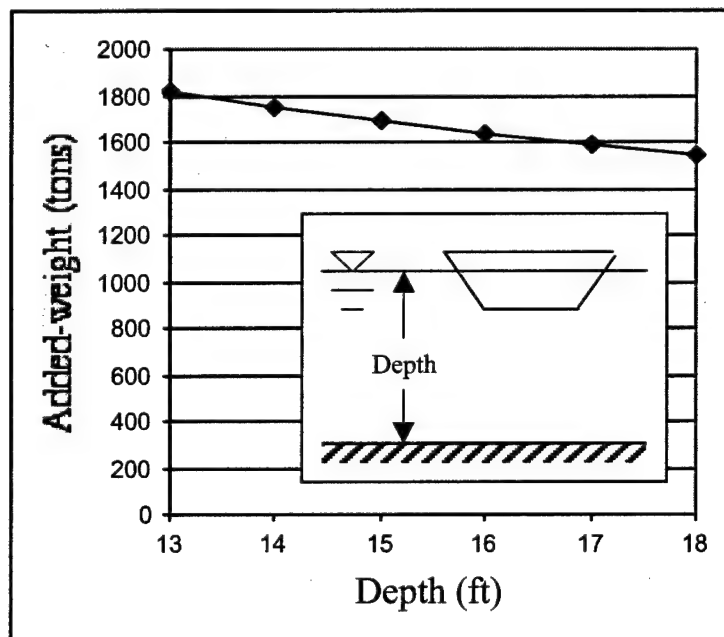


Figure 5. Water-depth effect on added weight of loaded JOHB traveling parallel to wall at a speed of 3 fps with a rake angle of 45 deg

Effect of Barge-Wall Separation and Barge Load

Because of the rake angle of the bow, the only separation effect occurs when the barge travels parallel to the quay wall. The wall effect extends only several feet from the wall, as can be seen in the results presented in Figure 6. Hence, for all but the smallest of approach angles (α), there is no wall effect, according to the theory. This is due to the low barge speed. The results are presented for the JOHB under both loaded and unloaded conditions.

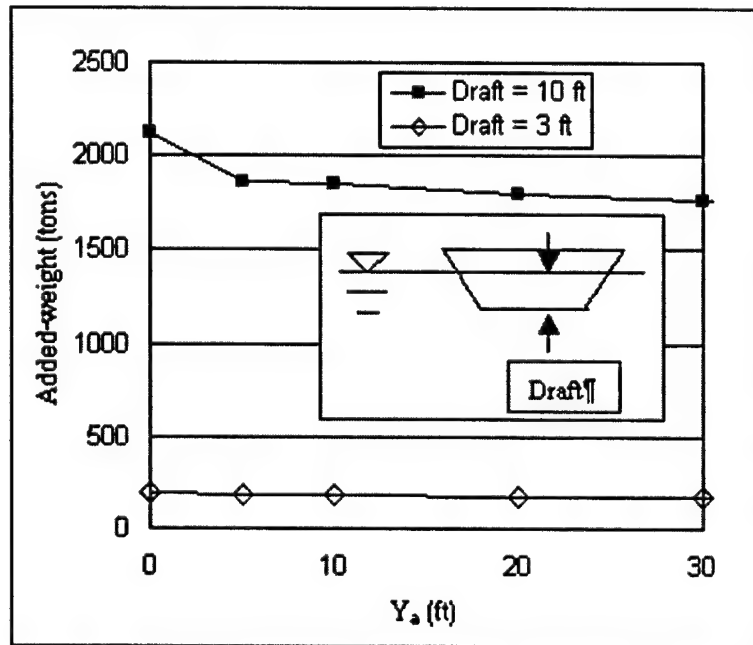


Figure 6. Wall-distance effect on the added weight of the loaded and unloaded JOHB traveling parallel to the wall in 18 ft of water at a speed of 3 fps with a rake angle of 45 deg (Here, Y_s is the distance between the wall-side front corner of the barge and the wall, measured perpendicular to the wall)

Rake Angle Effect

Although the rake angle of both the JOHB and the SOHB is fixed, the effect of rake angle is presented in Figure 7. The added weight is seen to be greatest for the barge with a plumb bow ($\theta = 90$ deg). The results shown in this figure demonstrate why barges at the front of a flotilla usually have rake angles less than 90 deg. The purpose of this exercise is to help barge designers determine an optimum configuration.

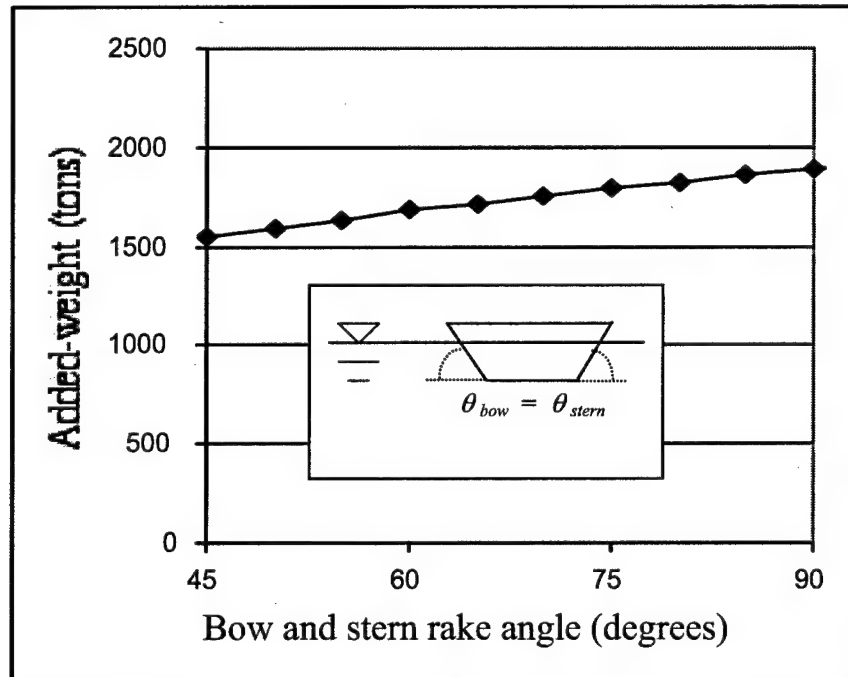


Figure 7. Effect of bow rake angle on added weight of loaded JOHB operating in 18-ft-deep open waters at a speed of 3 fps

Comparison of JOHB and SOHB

In this case, the JOHB and SOHB are compared under loaded conditions, operating in 18 ft of open water, at a speed of 3 fps. For the JOHB, the added weight is 1,548 tons; for the SOHB, the value is 1,179 tons (see Appendix B). Although the scale factors for the two barges differ for both length and width, the behaviors shown in Figures 4-7 can be expected to be similar for the SOHB.

3 Discussion and Conclusions

During Phase 1 of this study, a box-lighter barge model is towed in the wave/towing tank at The Johns Hopkins University, as discussed in McCormick et al. (2002). The barge model spanned the tank, approximating a barge of infinite beam (2-D). The barge is accelerated perpendicular to the wall at the end of the tank using a gravity towing system; a digital camera is used to study the free surface. Complicated dynamics are observed as the barge approaches the wall, prompting further study. An approximate solution of the potential-flow problem of the barge in proximity to the wall is formulated by assuming the free-surface profile observed during the towing tests. Because the tests approximated 2-D flow, a behavior of free surface in the third dimension (in the direction of the beam) is assumed. Then, the added mass is calculated from empirical measurements of the free surface, not directly. This analysis has significant limitations, and results were limited to a maximum barge speed of 4 fps. A less-restrictive, fully 3-D solution to the problem is pursued in the present analysis, Phase 2.

The objective of the present study is to numerically model the added mass of a barge operating in restricted waters, i.e., waters of finite depth that are bounded by a vertical wall. To model the hydrodynamics as a potential-flow problem, it is assumed that the fluid flow is incompressible and irrotational. Furthermore, the model is based on the assumption of a low Froude number (see Equation 6). The barge speed is low enough that the surface waves are small in comparison to the barge length. This low-Froude number assumption would be invalid for calculations of radiation damping, for instance, which is highly dependent on speed. However, for the speeds likely to be experienced by the barges of interest (7 fps at the absolute maximum), this assumption is believed to be valid for added-mass calculations. A restriction that the Froude number be less than 0.1 ensures that the wave drag and, thus, the surface waves are negligible, as seen for example in Comstock (1967). This restriction corresponds to a maximum barge speed of 8 fps.

In an attempt to assess the numerical added-mass calculations, comparisons were made with experimental and analytical results presented in Garrison (1974). Most added-mass calculations deal with the hydrodynamics of ships oscillating in water waves; few applications involve nonoscillatory motion, as in the present study. For the simplified geometry of a hemisphere oscillating in water, Garrison

shows that the normalized added mass $[m_w / (\rho r^3)]$, where ρ is the mass density of the fluid and r is the radius of the hemisphere] goes to $\pi / 3$ as the frequency of oscillation goes to zero. Numerical calculations are shown to give the same result. The program designed for barge added-mass calculations was modified to calculate the added mass of a hemisphere, for comparison with these analytical and numerical results. The modified program code is shown in Appendix A (Section 3). Appendix B (Section 3) shows a sample calculation of the added mass of a hemisphere. The normalized added mass is found to be 0.81, a difference of 23 percent from the analytical and numerical results presented by Garrison (1974).

The numerical solution to the problem allows for the body geometry to be arbitrary. Currently, the FORTRAN code restricts the barge to having vertical sides and a flat bottom; the bow and stern rake angles are variable. The length, draft, beam, and angles are all parameters that may be varied. For the JOHB and SOHB, the program discretizes the body surface into approximately 60 panels. This number is found to be a good compromise between convergence of the solution and computational efficiency. If barges of significantly different geometry are to be studied, the FORTRAN code should be modified to find the optimal number of panels.

The predicted effect of the barge speed on the added weight can be seen in Figure 4 for the JOHB operating in open waters where the water depth is 18 ft. With the low-Froude number assumption, the added weight is essentially constant in the speed regime considered here. This seems counterintuitive, but physically, it means that the barge is accelerating roughly the same amount of water around it as it moves. The kinetic energy of that accelerated water is much greater, since that is proportional to the square of the velocity. Furthermore, this result implies that if one were to accelerate (or decelerate) the barge, the water would exert an inertial force on the barge that is proportional to the acceleration, not the velocity.

The effects of water depth on the added weight are shown in Figure 5 for the JOHB traveling at 3 fps in 18-ft-deep open waters. For the depths studied, the added weight is seen to decrease with depth, in disagreement with the Phase 1 result. This is more intuitive, since as the depth approaches the draft of the barge, the barge must accelerate nearly all of the water in front of and behind it.

In Figure 6, the effect of the separation distance (Y_a) between the quay wall and the JOHB traveling parallel to the wall is shown. It can be seen in that figure that the wall effect at a 3-fps barge speed is observed only in close proximity to the wall, for both loaded and unloaded conditions.

The effect of varying the rake angle (θ) of the bow and stern of the JOHB is seen in Figure 7. As expected, the added mass is smaller for a raked barge ($\theta_{bow} = \theta_{stern} = 45$ deg) than for a box-lighter ($\theta_{bow} = \theta_{stern} = 90$ deg).

The added weight values presented here are orders of magnitude greater than those presented in the Phase 1 report. For example, for the JOHB traveling in 18-ft-deep open waters at 3 fps, the added weight is 77 percent of the barge's weight

from the Phase 2 calculations, as opposed to 0.2 percent from the Phase 1 calculations. Physically, this means that, in the first case, the barge's motion accelerates a volume of water equal to 77 percent of the barge's displacement volume. This value is in line with previous experimental results. The Phase 2 analysis is considered superior to the Phase 1 analysis, as expected, because it solves the complete boundary-value problem without using empirical data.

The analysis presented herein has significant assumptions, but these assumptions are believed to be reasonable for the design conditions. The analysis presented herein is considered to be superior to the potential theories that were based on 2-D models, as reported by Wendel (1956) and Brennen (1982). The numerical analysis of Phase 2 allows for arbitrary barge geometries, and does not rely on empirical results, as did the Phase 1 analysis.

There is no theoretical hurdle in the way of extending the present analysis to a flotilla of barges. With some work, the program code could be modified to distribute panels over several barges, including their reflections. A practical consideration that may prove troublesome would be the computing demands of such an analysis. However, it is possible to make use of planes of symmetry to reduce the number of computations.

References

- Brennen, C. E. (1982), "A review of added mass and fluid inertial forces," Report N62583-81-MR-554, Naval Civil Engineering Laboratory, Port Hueneme, CA.
- Comstock, J. P., ed. (1967). *Principles of naval architecture*. Society of Naval Architects and Marine Engineers, New York. 301-12.
- Garrison, C. J. (1974), "Hydrodynamics of large objects in the sea; Part I, Hydrodynamic analysis," *Journal of Hydronautics* 8(1).
- Karamcheti, K. (1966), *Principles of ideal-fluid aerodynamics*. John Wiley and Sons, New York.
- McCormick, M. E., Kraemer, D. R. B., Hudson, P. and Noble, W. (2002). "Analysis of the added mass of a barge in restricted waters; Phase 1 model," Technical Report ERDC/ITL TR-02-8, U.S. Army Engineer Research and Development Center, Vicksburg, MS.
- Wendel, Kurt. (1956). "Hydrodynamic masses and hydrodynamic moments of inertia," David Taylor Model Basin Translation 260, Naval Surface Warfare Center, Carderock Division, Maryland.

Appendix A

FORTRAN Computer Code

1. WESbargeBEM: Barge added-mass calculation program

```

C by D. R. B. Kraemer
C Calculates the added mass of a barge in proximity to a wall
C using a boundary-element potential-flow solution.
C Note that the parameter xElements is set to give the
C optimal number of panels for the Jumbo Open-Hopper Barge
C (JOHB) or the Standard Open-Hopper Barge (SOHB). For other
C geometries, this parameter should be re-optimized.

INTEGER maxElements
PARAMETER (maxElements= 1000)
C ***** NOTE: Change this ^^^^^ parameter if the xElements parameter
C ***** below is changed
REAL pi, rho, g, degrees2radians, feet2meters, kg2tons
REAL length, beam, draft, thetaBow, thetaStern
REAL depth, speed, alpha, wallDistance
REAL elementSize, deltaX,
  & deltaY, deltaZ, nodes(maxElements*4, 3)
INTEGER elements, i, j, k, p, q, xElements, yElements, zElements,
  & integralSteps, bargeElements
REAL x, y, z, dummy
REAL sidel(maxElements,3), side2(maxElements,3), cross(3),
  & area(maxElements), n(maxElements,3), centroid(maxElements,3)
DOUBLE PRECISION influence(maxElements, maxElements),
  & influenceCopy(maxElements,maxElements),
  & rhs(maxElements), source(maxElements), phi
DOUBLE PRECISION rij(3), radius, point(3), rPoint(3),
  & pointArea
REAL addedMass, bargeMass

C ***** Constants
pi= 4. *ATAN(1.)
rho= 1000. ! water mass density, kg/m^3
g= 9.81 ! acceleration due to gravity, m/s^2
degrees2radians= pi /180. ! conversion factor
feet2meters= 12. *.0254 ! conversion factor
kg2tons= 2.205 /2000. ! conversion factor

```

```

C ***** Barge input
1 WRITE(*,*) 'Enter barge wetted length, ft:'
  READ(*,*) length
  length=length *feet2meters
  WRITE(*,*) 'Enter barge wetted beam, ft:'
  READ(*,*) beam
  beam=beam *feet2meters
  WRITE(*,*) 'Enter barge draft, ft:'
  READ(*,*) draft
  draft=draft *feet2meters
  WRITE(*,*) 'Enter barge bow rake angle, degrees (90 = vertical):'
  READ(*,*) thetaBow
  thetaBow=thetaBow *degrees2radians
  WRITE(*,*) 'Enter barge stern rake angle, degrees (90 = vertical):'
  READ(*,*) thetaStern
  thetaStern=thetaStern *degrees2radians
  WRITE(*,*) 'Enter water depth, ft:'
  READ(*,*) depth
  depth= depth *feet2meters
  WRITE(*,*) 'Enter barge speed, ft/s:'
  READ(*,*) speed
  speed= speed *feet2meters
  WRITE(*,*) 'Enter barge distance from wall, ft:'
  READ(*,*) wallDistance
  wallDistance= wallDistance *feet2meters
  WRITE(*,*) 'Enter barge angle to wall, degrees:'
  WRITE(*,*) '(0 = parallel, 90 = perpendicular)'
  READ(*,*) alpha
  alpha= alpha *degrees2radians
c WRITE(*,*) 'Enter approximate # of elements along the length:'
c READ(*,*) xElements
C ***** Note: xElements is set to 5. This provides an optimal
C number of elements for the JOHB and SOHB. For other barge
C geometries, xElements should be re-optimized.
  xElements= 5
  elementSize= (length -draft/TAN((thetaBow+thetaStern)/2.) ) /
    & REAL(xElements)

  yElements= NINT(beam /elementSize)
  IF (yElements .LT. 1) yElements= 1
  deltaY= beam /yElements
  zElements= NINT(draft /elementSize)
  IF (zElements .LT. 1) zElements= 1
  deltaZ= draft /zElements

  elements= 0

C ***** +y side
  y= beam /2.
  z= 0.

  DO 10 j=1, zElements
    x= -length /2. + (-z +deltaZ) / TAN(thetaStern)
    deltaX= (length -2. *(-z +deltaZ) /

```

```

&          TAN((thetaBow+thetaStern)/2.) )/ xElements

DO 20 i= 1, xElements
  nodes(4*elements +1,1)= x
  nodes(4*elements +1,2)= y
  nodes(4*elements +1,3)= z

  nodes(4*elements +2,1)= x +deltaX
  nodes(4*elements +2,2)= y
  nodes(4*elements +2,3)= z

  nodes(4*elements +3,1)= x +deltaX
  nodes(4*elements +3,2)= y
  nodes(4*elements +3,3)= z -deltaZ
  nodes(4*elements +4,1)= x
  nodes(4*elements +4,2)= y
  nodes(4*elements +4,3)= z -deltaZ

  elements= elements +1
  x= x +deltaX
20  CONTINUE

  z= z -deltaZ

10  CONTINUE

C ***** +x side
deltaX= deltaZ /TAN((thetaBow+thetaStern)/2.)
x= length /2.
z= 0.

DO 30 j=1, zElements
  y= beam /2.

DO 40 i= 1, xElements
  nodes(4*elements +1,1)= x
  nodes(4*elements +1,2)= y
  nodes(4*elements +1,3)= z

  nodes(4*elements +2,1)= x
  nodes(4*elements +2,2)= y -deltaY
  nodes(4*elements +2,3)= z

  nodes(4*elements +3,1)= x -deltaX
  nodes(4*elements +3,2)= y -deltaY
  nodes(4*elements +3,3)= z -deltaZ

  nodes(4*elements +4,1)= x -deltaX
  nodes(4*elements +4,2)= y
  nodes(4*elements +4,3)= z -deltaZ

  elements= elements +1
  y= y -deltaY
40  CONTINUE

```

```

        x= x -deltaX
        z= z -deltaZ

30      CONTINUE

C ***** -y side
y= -beam /2.
z= 0.

DO 50 j=1, zElements
    x= length /2. - (-z +deltaZ) /TAN(thetaBow)
    deltaX= (length -2. *(-z +deltaZ) /
&          TAN((thetaBow+thetaStern)/2.) )/ xElements

    DO 60 i= 1, xElements
        nodes(4*elements +1,1)= x
        nodes(4*elements +1,2)= y
        nodes(4*elements +1,3)= z

        nodes(4*elements +2,1)= x -deltaX
        nodes(4*elements +2,2)= y
        nodes(4*elements +2,3)= z

        nodes(4*elements +3,1)= x -deltaX
        nodes(4*elements +3,2)= y
        nodes(4*elements +3,3)= z -deltaZ

        nodes(4*elements +4,1)= x
        nodes(4*elements +4,2)= y
        nodes(4*elements +4,3)= z -deltaZ

        elements= elements +1
        x= x -deltaX
60      CONTINUE

        z= z -deltaZ

50      CONTINUE

C ***** -x side
deltaX= deltaZ /TAN((thetaBow+thetaStern)/2.)
x= -length /2.
z= 0.

DO 70 j=1, zElements
    y= -beam /2.

    DO 80 i= 1, xElements
        nodes(4*elements +1,1)= x
        nodes(4*elements +1,2)= y
        nodes(4*elements +1,3)= z

        nodes(4*elements +2,1)= x

```

```

        nodes(4*elements +2,2)= y +deltaY
        nodes(4*elements +2,3)= z

        nodes(4*elements +3,1)= x +deltaX
        nodes(4*elements +3,2)= y +deltaY
        nodes(4*elements +3,3)= z -deltaZ

        nodes(4*elements +4,1)= x +deltaX
        nodes(4*elements +4,2)= y
        nodes(4*elements +4,3)= z -deltaZ

        elements= elements +1
        y= y +deltaY
80      CONTINUE
        0
        x= x +deltaX
        z= z -deltaZ

70      CONTINUE

C ***** -z side
y= beam /2.
z= -draft
deltaX= ( length -2. *draft /TAN(thetaBow) ) /xElements

DO 90 j=1, zElements
    x= -length /2. + draft /TAN(thetaStern)

    DO 100 i= 1, xElements
        nodes(4*elements +1,1)= x
        nodes(4*elements +1,2)= y
        nodes(4*elements +1,3)= z

        nodes(4*elements +2,1)= x +deltaX
        nodes(4*elements +2,2)= y
        nodes(4*elements +2,3)= z

        nodes(4*elements +3,1)= x +deltaX
        nodes(4*elements +3,2)= y -deltaY
        nodes(4*elements +3,3)= z

        nodes(4*elements +4,1)= x
        nodes(4*elements +4,2)= y -deltaY
        nodes(4*elements +4,3)= z

        elements= elements +1
        x= x +deltaX
100     CONTINUE

        y= y -deltaY

90      CONTINUE

```

```

C ***** Move mesh to position
IF (alpha .EQ. 0.) THEN
    DO 110 i= 1, 4 *elements
        nodes(i,1)= nodes(i,1) -length /2.
        nodes(i,2)= nodes(i,2) +beam /2. +wallDistance
110        CONTINUE

    ELSE

        DO 120 i= 1, 4 *elements
            dummy= nodes(i,1) *COS(alpha) +nodes(i,2) *SIN(alpha) -
            &         length /2. *COS(alpha) +beam /2. *SIN(alpha) -
            &         wallDistance /TAN(alpha)
            nodes(i,2)= nodes(i,2) *COS(alpha) -
            &         nodes(i,1)*SIN(alpha) + beam /2. *COS(alpha) +
            &         length /2. *SIN(alpha) + wallDistance
            nodes(i,1)= dummy
120        CONTINUE

    ENDIF

    bargeElements= elements          ! number of elements on physical barge

c if (elements .GT. 1) go to 999    ! no reflections

C ***** Reflect barge: Surface
DO 130 i= 1, elements
    nodes(4*elements +4*(i-1) +1,1)= nodes(4*(i-1)+4,1)
    nodes(4*elements +4*(i-1) +1,2)= nodes(4*(i-1)+4,2)
    nodes(4*elements +4*(i-1) +1,3)= -nodes(4*(i-1)+4,3)

    nodes(4*elements +4*(i-1) +2,1)= nodes(4*(i-1)+3,1)
    nodes(4*elements +4*(i-1) +2,2)= nodes(4*(i-1)+3,2)
    nodes(4*elements +4*(i-1) +2,3)= -nodes(4*(i-1)+3,3)

    nodes(4*elements +4*(i-1) +3,1)= nodes(4*(i-1)+2,1)
    nodes(4*elements +4*(i-1) +3,2)= nodes(4*(i-1)+2,2)
    nodes(4*elements +4*(i-1) +3,3)= -nodes(4*(i-1)+2,3)

    nodes(4*elements +4*(i-1) +4,1)= nodes(4*(i-1)+1,1)
    nodes(4*elements +4*(i-1) +4,2)= nodes(4*(i-1)+1,2)
    nodes(4*elements +4*(i-1) +4,3)= -nodes(4*(i-1)+1,3)
130    CONTINUE
    elements= 2 *elements

C ***** Reflect barge: Bottom
DO 400 i= 1, elements
    nodes(4*elements +4*(i-1) +1,1)= nodes(4*(i-1)+1,1)
    nodes(4*elements +4*(i-1) +1,2)= nodes(4*(i-1)+1,2)
    nodes(4*elements +4*(i-1) +1,3)= nodes(4*(i-1)+1,3) -
    &         2. *depth

    nodes(4*elements +4*(i-1) +2,1)= nodes(4*(i-1)+2,1)
    nodes(4*elements +4*(i-1) +2,2)= nodes(4*(i-1)+2,2)

```

```

        nodes(4*elements +4*(i-1) +2,3)= nodes(4*(i-1)+2,3) -
&         2. *depth

        nodes(4*elements +4*(i-1) +3,1)= nodes(4*(i-1)+3,1)
        nodes(4*elements +4*(i-1) +3,2)= nodes(4*(i-1)+3,2)
        nodes(4*elements +4*(i-1) +3,3)= nodes(4*(i-1)+3,3) -
&         2. *depth

        nodes(4*elements +4*(i-1) +4,1)= nodes(4*(i-1)+4,1)
        nodes(4*elements +4*(i-1) +4,2)= nodes(4*(i-1)+4,2)
        nodes(4*elements +4*(i-1) +4,3)= nodes(4*(i-1)+4,3) -
&         2. *depth
400    CONTINUE
        elements= 2 *elements

C ***** Reflect barge: Bottom-Surface reflection
DO 410 i= 1, elements
    nodes(4*elements +4*(i-1) +1,1)= nodes(4*(i-1)+1,1)
    nodes(4*elements +4*(i-1) +1,2)= nodes(4*(i-1)+1,2)
    nodes(4*elements +4*(i-1) +1,3)= nodes(4*(i-1)+1,3) +
&         2. *depth

    nodes(4*elements +4*(i-1) +2,1)= nodes(4*(i-1)+2,1)
    nodes(4*elements +4*(i-1) +2,2)= nodes(4*(i-1)+2,2)
    nodes(4*elements +4*(i-1) +2,3)= nodes(4*(i-1)+2,3) +
&         2. *depth

    nodes(4*elements +4*(i-1) +3,1)= nodes(4*(i-1)+3,1)
    nodes(4*elements +4*(i-1) +3,2)= nodes(4*(i-1)+3,2)
    nodes(4*elements +4*(i-1) +3,3)= nodes(4*(i-1)+3,3) +
&         2. *depth

    nodes(4*elements +4*(i-1) +4,1)= nodes(4*(i-1)+4,1)
    nodes(4*elements +4*(i-1) +4,2)= nodes(4*(i-1)+4,2)
    nodes(4*elements +4*(i-1) +4,3)= nodes(4*(i-1)+4,3) +
&         2. *depth
410    CONTINUE
        elements= 3 *(elements /2)

C ***** Reflect barge: Wall reflection
DO 420 i= 1, elements
    nodes(4*elements +4*(i-1) +1,1)= nodes(4*(i-1)+1,1)
    nodes(4*elements +4*(i-1) +1,2)= -nodes(4*(i-1)+1,2)
    nodes(4*elements +4*(i-1) +1,3)= nodes(4*(i-1)+1,3)

    nodes(4*elements +4*(i-1) +2,1)= nodes(4*(i-1)+2,1)
    nodes(4*elements +4*(i-1) +2,2)= -nodes(4*(i-1)+2,2)
    nodes(4*elements +4*(i-1) +2,3)= nodes(4*(i-1)+2,3)

    nodes(4*elements +4*(i-1) +3,1)= nodes(4*(i-1)+3,1)
    nodes(4*elements +4*(i-1) +3,2)= -nodes(4*(i-1)+3,2)
    nodes(4*elements +4*(i-1) +3,3)= nodes(4*(i-1)+3,3)

    nodes(4*elements +4*(i-1) +4,1)= nodes(4*(i-1)+4,1)

```

```

        nodes(4*elements +4*(i-1) +4,2)= -nodes(4*(i-1)+4,2)
        nodes(4*elements +4*(i-1) +4,3)= nodes(4*(i-1)+4,3)
420    CONTINUE
        elements= 2 *elements

c999    continue

        IF (elements .GT. maxElements) THEN
            WRITE(*,*) 'Too many elements in the mesh!'
            WRITE(*,*) 'Reduce number of elements, or increase'
            WRITE(*,*) 'the parameter maxElements in the code.'
            GO TO 1
        ENDIF

c do 99 i= 1, 4*elements
c      write(*,*) (nodes(i,j), j= 1, 3)
c99    continue

C ***** Calculate element centroids, areas, & normal vectors
DO 140 i= 1, elements
    DO 150 j= 1, 3
        side1(i,j)= nodes(4*(i-1) +2, j) -nodes(4*(i-1) +1, j)
        side2(i,j)= nodes(4*(i-1) +4, j) -nodes(4*(i-1) +1, j)
150    CONTINUE

C ***** area is the magnitude of side1 cross side2.
        cross(1)= side1(i,2) *side2(i,3) -side1(i,3) *side2(i,2)
        cross(2)= side1(i,3) *side2(i,1) -side1(i,1) *side2(i,3)
        cross(3)= side1(i,1) *side2(i,2) -side1(i,2) *side2(i,1)

        area(i)= SQRT( (cross(1))**2 +(cross(2))**2 +
            & (cross(3))**2 )
c write(*,*) 'area ', i, '=', area(i)

        DO 160 j= 1, 3
            n(i,j)= cross(j) /area(i)

            centroid(i,j)= 0.
            DO 170 k= 1, 4
                centroid(i,j)= centroid(i,j) +nodes(4*(i-1)+k,j) /
                    & 4.
170            CONTINUE

160        CONTINUE

140    CONTINUE

        write(*,*) 'Total elements:', elements
c write(*,*) 'areas'
c do 87 i= 1, elements
c      write(*,*) area(i)
c87    continue
c write(*,*) 'centroids'
c do 88 i= 1, elements

```

```

c      write(*,*) (centroid(i,j), j= 1, 3)
c88    continue
c  write(*,*) 'normals'
c  do 89 i= 1, elements
c      write(*,*) (n(i,j), j= 1, 3)
c89    continue

C ***** Calculate influence matrix
integralSteps= 10

WRITE(*,*) 'Calculating influence matrix:'
DO 200 j= 1, elements
  IF ( ( REAL(j) / REAL(NINT( REAL(elements)/10. )) -
    & REAL( INT( REAL(j) / REAL(NINT( REAL(elements)/10.)) ) ) )
    & .EQ. 0. ) WRITE(*,*) NINT( REAL(100 *j)/ REAL(elements) ),
    & '% done...'

  DO 210 i= 1, elements
    influence(j,i)= 0.

    IF (i .EQ. j) THEN
      influence(j,i)= 2. *pi
    ELSE
      DO 220 k= 1, 3
        rij(k)= centroid(j,k) -centroid(i,k)
220      CONTINUE
        radius= DSQRT( (rij(1))**2 +(rij(2))**2 +
          & (rij(3))**2 )

        IF (REAL(radius) /elementSize .GE. 5. ) THEN
          influence(j,i)= 1. /radius /radius *
          & ( rij(1) *n(j,1) +rij(2) *n(j,2) +rij(3) *n(j,3) ) /
          & radius *area(i)
        ELSE
          pointArea= area(i) /REAL(integralSteps) /
          & REAL(integralSteps)

          DO 230 p= 0, integralSteps -1
            DO 240 q= 0, integralSteps -1

              DO 250 k= 1, 3
                point(k)= centroid(i,k) - sidel(i,k) /2. +
          & sidel(i,k) /integralSteps /2. + REAL(p) *sidel(i,k) /
          & integralSteps - side2(i,k) /2. +
          & side2(i,k) /integralSteps /2. + REAL(q) *side2(i,k) /
          & integralSteps
                rPoint(k)= centroid(j,k) -point(k)
250      CONTINUE

                radius= DSQRT( (rPoint(1))**2 +
          & (rPoint(2))**2 + (rPoint(3))**2 )

                influence(j,i)= influence(j,i) +
          & 1. /radius /radius *

```

```

&    ( rPoint(1) *n(j,1) +rPoint(2) *n(j,2) +rPoint(3) *n(j,3) ) /
&    radius *pointArea

240          CONTINUE
230          CONTINUE

        ENDIF

        ENDIF

210          CONTINUE
200          CONTINUE

c  write(*,*) 'rhs vector:'
  DO 260 j= 1, elements
    rhs(j)= speed *COS(alpha) *n(j,1) -
      &      speed *SIN(alpha) *n(j,2)
c    write(*,*) REAL(rhs(j))
260    CONTINUE

c  WRITE(*,*) 'influence matrix:'
c  DO 263 i= 1, elements
c    WRITE(*,*) ( REAL(influence(i,j)), j=1,elements )
c 263    CONTINUE

    DO 232 i= 1, elements
      DO 234 j= 1, elements
        influenceCopy(i,j)= influence(i,j)
234      CONTINUE
232    CONTINUE

    WRITE(*,*) 'Calculating inverse of matrix.'

    CALL gaussDBLE(maxElements, elements,
      &      influenceCopy, rhs, source)

c  write(*,*) 'Source solution:'
c  do 265 i= 1, elements
c    write(*,*) REAL(source(i))
c265    continue

C  ***** Check Gaussian elimination calculation
c  WRITE(*,*) 'This should equal rhs above!'
c  DO 270 i= 1, elements
c    dummy= 0.
c    DO 280 j= 1, elements
c      dummy= dummy + influence(i,j) *source(j)
c280    CONTINUE
c    WRITE(*,*) dummy
c270    CONTINUE

C  ***** Calculate added mass

```

```

addedMass= 0.

WRITE(*,*) 'Calculating added mass:'

DO 300 j= 1, bargeElements
  IF ( ( REAL(j) / REAL(NINT( REAL(bargeElements)/10. )) -
& REAL( INT( REAL(j) / REAL(NINT( REAL(bargeElements)/10.)) ) ) )
& .EQ. 0. ) WRITE(*,*) NINT( REAL(100 *j)/ REAL(bargeElements) ),
& '% done...'

  phi= 0.

  DO 310 i= 1, elements

    DO 320 k= 1, 3
      rij(k)= centroid(j,k) -centroid(i,k)
320      CONTINUE
      radius= SQRT( (rij(1))**2 +(rij(2))**2 +
& (rij(3))**2 )

      IF (radius /elementSize .GE. 5. ) THEN
        phi= phi + source(i) /radius *area(i)
      ELSE
        pointArea= area(i) /REAL(integralSteps) /
& REAL(integralSteps)

        DO 330 p= 0, integralSteps -1

          DO 340 q= 0, integralSteps -1

            DO 350 k= 1, 3
              point(k)= centroid(i,k) - sidel(i,k) /2. +
& sidel(i,k) /integralSteps /2. + REAL(p) *sidel(i,k) /
& integralSteps - side2(i,k) /2. +
& side2(i,k) /integralSteps /2. + REAL(q) *side2(i,k) /
& integralSteps
              rPoint(k)= centroid(j,k) -point(k)
350              CONTINUE

              radius= SQRT( (rPoint(1))**2 +(rPoint(2))**2 +
& (rPoint(3))**2 )

              phi= phi + source(i) /radius *pointArea

340              CONTINUE
330              CONTINUE
            ENDIF

310            CONTINUE
          addedMass= addedMass + phi *rhs(j) *area(j)
300          CONTINUE

    addedMass= addedMass *rho /speed /speed
    WRITE(*,*) 'Added-mass =', addedMass, ' kg,'

```

```
WRITE(*,*) addedMass *kg2tons, ' tons.'  
bargeMass= rho *( length - draft /  
    & TAN((thetaBow+thetaStern)/2.) ) *beam *draft  
WRITE(*,*) 'Added-mass / barge mass =', addedMass /bargeMass  
  
STOP  
END
```

2. WESSsphereBEM: Hemisphere added-mass calculation program

```

PROGRAM WESSsphereBEM
C by D. R. B. Kraemer
C Calculates the added mass of a barge in proximity to a wall
C using a boundary-element potential-flow solution.
C Note that the parameter xElements is set to give the
C optimal number of panels for the Jumbo Open-Hopper Barge
C (JOHB) or the Standard Open-Hopper Barge (SOHB). For other
C geometries, this parameter should be re-optimized.

INTEGER maxElements
PARAMETER (maxElements= 1000)
REAL pi, rho, g, degrees2radians, feet2meters, kg2tons
REAL r, dummy
REAL depth, speed, alpha, wallDistance
REAL theta, deltaTheta, rho1, rho2, arc, beta, deltaBeta, betaElements
REAL z1, z2
REAL nodes(maxElements*4, 3)
INTEGER elements, i, j, k, p, q, zElements,
    & integralSteps, bargeElements, wallFlag
REAL elementSize
REAL sidel(maxElements,3), side2(maxElements,3), cross(3),
    & area(maxElements), n(maxElements,3), centroid(maxElements,3)
DOUBLE PRECISION influence(maxElements, maxElements),
    & influenceCopy(maxElements,maxElements),
    & rhs(maxElements), rhscopy(maxElements), source(maxElements), phi
DOUBLE PRECISION rij(3), radius, point(3), rPoint(3),
    & pointArea
REAL addedMass, bargeMass

C ***** Constants
pi= 4. *ATAN(1.)
rho= 1000. ! water mass density, kg/m^3
g= 9.81 ! acceleration due to gravity, m/s^2
degrees2radians= pi /180. ! conversion factor
feet2meters= 12. *.0254 ! conversion factor
kg2tons= 2.205 /2000. ! conversion factor

C ***** Barge input
WRITE(*,*) 'Enter hemisphere radius, ft:'
READ(*,*) r
r=r *feet2meters

WRITE(*,*) 'Enter barge speed, ft/s:'
READ(*,*) speed
speed= speed *feet2meters
WRITE(*,*) 'Do you want to consider wall and bottom distances?'
WRITE(*,*) 'Enter 1 for yes, 0 for no:'
READ(*,*) wallFlag
IF (wallFlag .EQ. 0) GO TO 1
WRITE(*,*) 'Enter water depth, ft:'
READ(*,*) depth
depth= depth *feet2meters

```

```

WRITE(*,*) 'Enter barge distance from wall, ft:'
READ(*,*) wallDistance
wallDistance= wallDistance *feet2meters
WRITE(*,*) 'Enter barge angle to wall, degrees:'
WRITE(*,*) '(0 = parallel, 90 = perpendicular)'
READ(*,*) alpha
alpha= alpha *degrees2radians
1 WRITE(*,*) 'Enter approximate # of elements along the draft:'
READ(*,*) zElements
c ***** Note: zElements is set to 5. This provides an optimal
C number of elements for the JOHB and SOHB. For other barge
C geometries, xElements should be re-optimized.
c zElements= 5

elements= 0

C *****

deltaTheta= pi /2. / zElements

arc= r *deltaTheta
elementSize= arc

theta= pi /1000.      ! approximately zero

DO 10 j=1, zElements
c write(*,*) 'theta =', theta

    rho1= r *SIN(theta)
    rho2= r *SIN(theta +deltaTheta)

    betaElements= NINT( 2. *pi *rho2 /arc )
    deltaBeta= 2. *pi /betaElements

    z1= -r *COS(theta)
    z2= -r *COS(theta +deltaTheta)
    beta= 0.

    DO 20 i= 1, betaElements

C      ***** for spherical element areas:
        area(elements+1)= (rho1 +rho2)/2. *deltaBeta *r *deltaTheta

c write(*,*) 'beta =', beta

        nodes(4*elements +1,1)= rho1 *COS(beta)
        nodes(4*elements +1,2)= rho1 *SIN(beta)
        nodes(4*elements +1,3)= z1

        nodes(4*elements +2,1)= rho1 *COS(beta +deltaBeta)
        nodes(4*elements +2,2)= rho1 *SIN(beta +deltaBeta)
        nodes(4*elements +2,3)= z1

```

```

        nodes(4*elements +3,1)= rho2 *COS(beta +deltaBeta)
        nodes(4*elements +3,2)= rho2 *SIN(beta +deltaBeta)
        nodes(4*elements +3,3)= z2

        nodes(4*elements +4,1)= rho2 *COS(beta)
        nodes(4*elements +4,2)= rho2 *SIN(beta)
        nodes(4*elements +4,3)= z2

        elements= elements +1
        beta= beta +deltaBeta
20      CONTINUE

        theta= theta +deltaTheta

10      CONTINUE

        bargeElements= elements

C ***** Reflect barge: Surface
DO 130 i= 1, elements
    area(elements +i)= area(i)
    nodes(4*elements +4*(i-1) +1,1)= nodes(4*(i-1)+4,1)
    nodes(4*elements +4*(i-1) +1,2)= nodes(4*(i-1)+4,2)
    nodes(4*elements +4*(i-1) +1,3)= -nodes(4*(i-1)+4,3)

    nodes(4*elements +4*(i-1) +2,1)= nodes(4*(i-1)+3,1)
    nodes(4*elements +4*(i-1) +2,2)= nodes(4*(i-1)+3,2)
    nodes(4*elements +4*(i-1) +2,3)= -nodes(4*(i-1)+3,3)

    nodes(4*elements +4*(i-1) +3,1)= nodes(4*(i-1)+2,1)
    nodes(4*elements +4*(i-1) +3,2)= nodes(4*(i-1)+2,2)
    nodes(4*elements +4*(i-1) +3,3)= -nodes(4*(i-1)+2,3)

    nodes(4*elements +4*(i-1) +4,1)= nodes(4*(i-1)+1,1)
    nodes(4*elements +4*(i-1) +4,2)= nodes(4*(i-1)+1,2)
    nodes(4*elements +4*(i-1) +4,3)= -nodes(4*(i-1)+1,3)
130    CONTINUE
    elements= 2 *elements

    if (wallFlag .EQ. 0) go to 999 ! SKIP WALL/BOTTOM REFLECTIONS!!!!!!

C ***** Reflect barge: Bottom
DO 400 i= 1, elements
    area(elements +i)= area(i)
    nodes(4*elements +4*(i-1) +1,1)= nodes(4*(i-1)+1,1)
    nodes(4*elements +4*(i-1) +1,2)= nodes(4*(i-1)+1,2)
    nodes(4*elements +4*(i-1) +1,3)= nodes(4*(i-1)+1,3) -
&        2. *depth

    nodes(4*elements +4*(i-1) +2,1)= nodes(4*(i-1)+2,1)
    nodes(4*elements +4*(i-1) +2,2)= nodes(4*(i-1)+2,2)
    nodes(4*elements +4*(i-1) +2,3)= nodes(4*(i-1)+2,3) -
&        2. *depth

```

```

        nodes(4*elements +4*(i-1) +3,1)= nodes(4*(i-1)+3,1)
        nodes(4*elements +4*(i-1) +3,2)= nodes(4*(i-1)+3,2)
        nodes(4*elements +4*(i-1) +3,3)= nodes(4*(i-1)+3,3) -
&          2. *depth

        nodes(4*elements +4*(i-1) +4,1)= nodes(4*(i-1)+4,1)
        nodes(4*elements +4*(i-1) +4,2)= nodes(4*(i-1)+4,2)
        nodes(4*elements +4*(i-1) +4,3)= nodes(4*(i-1)+4,3) -
&          2. *depth
400    CONTINUE
        elements= 2 *elements

C ***** Reflect barge: Bottom-Surface reflection
DO 410 i= 1, elements
    area(elements +i)= area(i)
    nodes(4*elements +4*(i-1) +1,1)= nodes(4*(i-1)+1,1)
    nodes(4*elements +4*(i-1) +1,2)= nodes(4*(i-1)+1,2)
    nodes(4*elements +4*(i-1) +1,3)= nodes(4*(i-1)+1,3) +
&          2. *depth

    nodes(4*elements +4*(i-1) +2,1)= nodes(4*(i-1)+2,1)
    nodes(4*elements +4*(i-1) +2,2)= nodes(4*(i-1)+2,2)
    nodes(4*elements +4*(i-1) +2,3)= nodes(4*(i-1)+2,3) +
&          2. *depth

    nodes(4*elements +4*(i-1) +3,1)= nodes(4*(i-1)+3,1)
    nodes(4*elements +4*(i-1) +3,2)= nodes(4*(i-1)+3,2)
    nodes(4*elements +4*(i-1) +3,3)= nodes(4*(i-1)+3,3) +
&          2. *depth

    nodes(4*elements +4*(i-1) +4,1)= nodes(4*(i-1)+4,1)
    nodes(4*elements +4*(i-1) +4,2)= nodes(4*(i-1)+4,2)
    nodes(4*elements +4*(i-1) +4,3)= nodes(4*(i-1)+4,3) +
&          2. *depth
410    CONTINUE
        elements= 3 *(elements /2)

C ***** Reflect barge: Wall reflection
DO 420 i= 1, elements
    area(elements +i)= area(i)
    nodes(4*elements +4*(i-1) +1,1)= nodes(4*(i-1)+1,1)
    nodes(4*elements +4*(i-1) +1,2)= -nodes(4*(i-1)+1,2)
    nodes(4*elements +4*(i-1) +1,3)= nodes(4*(i-1)+1,3)

    nodes(4*elements +4*(i-1) +2,1)= nodes(4*(i-1)+2,1)
    nodes(4*elements +4*(i-1) +2,2)= -nodes(4*(i-1)+2,2)
    nodes(4*elements +4*(i-1) +2,3)= nodes(4*(i-1)+2,3)

    nodes(4*elements +4*(i-1) +3,1)= nodes(4*(i-1)+3,1)
    nodes(4*elements +4*(i-1) +3,2)= -nodes(4*(i-1)+3,2)
    nodes(4*elements +4*(i-1) +3,3)= nodes(4*(i-1)+3,3)

    nodes(4*elements +4*(i-1) +4,1)= nodes(4*(i-1)+4,1)

```

```

        nodes(4*elements +4*(i-1) +4,2)= -nodes(4*(i-1)+4,2)
        nodes(4*elements +4*(i-1) +4,3)= nodes(4*(i-1)+4,3)
420    CONTINUE
        elements= 2 *elements

999    continue

    IF (elements .GT. maxElements) THEN
        WRITE(*,*) 'Too many elements in the mesh!'
        WRITE(*,*) 'Reduce number of elements, or increase'
        WRITE(*,*) 'the parameter maxElements in the code.'
        GO TO 1
    ENDIF

c do 99 i= 1, 4*elements
c     write(*,*) (nodes(i,j), j= 1, 3)
c99    continue
C ***** Calculate element centroids, areas, & normal vectors
    DO 140 i= 1, elements
        DO 150 j= 1, 3
            side1(i,j)= nodes(4*(i-1) +2, j) -nodes(4*(i-1) +1, j)
            side2(i,j)= nodes(4*(i-1) +4, j) -nodes(4*(i-1) +1, j)
150    CONTINUE

C ***** area is the magnitude of side1 cross side2.
        cross(1)= side1(i,2) *side2(i,3) -side1(i,3) *side2(i,2)
        cross(2)= side1(i,3) *side2(i,1) -side1(i,1) *side2(i,3)
        cross(3)= side1(i,1) *side2(i,2) -side1(i,2) *side2(i,1)

        dummy= SQRT( (cross(1))**2 +(cross(2))**2 +
            & (cross(3))**2 )
c     write(*,*) 'area ', i, '=', area(i)

        DO 160 j= 1, 3
            n(i,j)= cross(j) /dummy

            centroid(i,j)= 0.
            DO 170 k= 1, 4
                centroid(i,j)= centroid(i,j) +nodes(4*(i-1)+k,j) /
                    & 4.
170    CONTINUE

160    CONTINUE

140    CONTINUE

        write(*,*) 'Total elements:', elements
c     write(*,*) 'areas'
c do 87 i= 1, elements
c     write(*,*) area(i)
c87    continue

        dummy= 0.
        do 87 i= 1, bargeElements

```

```

        dummy= dummy +area(i)
87 continue
    write(*,*) 'Calculated area, actual area:'
    write(*,*) dummy, 2. *pi *r *r

c  write(*,*) 'centroids'
c  do 88 i= 1, bargeElements
c      write(*,*) (centroid(i,j), j= 1, 3)
c88      continue
c  write(*,*) 'normals'
c  do 89 i= 1, bargeElements
c      write(*,*) (n(i,j), j= 1, 3)
c89      continue

C ***** Calculate influence matrix
    integralSteps= 10

    WRITE(*,*) 'Calculating influence matrix:'
    DO 200 j= 1, elements
        IF ( ( REAL(j) / REAL(NINT( REAL(elements)/10. )) -
& REAL( INT( REAL(j) / REAL(NINT( REAL(elements)/10.)) ) ) )
& .EQ. 0. ) WRITE(*,*) NINT( REAL(100 *j)/ REAL(elements) ),
& ' % done...'

        DO 210 i= 1, elements
            influence(j,i)= 0.

            IF (i .EQ. j) THEN
                influence(j,i)= 2. *pi
            ELSE
                DO 220 k= 1, 3
                    rij(k)= centroid(j,k) -centroid(i,k)
220                CONTINUE
                    radius= DSQRT( (rij(1))**2 +(rij(2))**2 +
& (rij(3))**2 )

                    IF (REAL(radius) /elementSize .GE. 5. ) THEN
                        influence(j,i)= 1. /radius /radius *
& ( rij(1) *n(j,1) +rij(2) *n(j,2) +rij(3) *n(j,3) ) /
& radius *area(i)
                    ELSE
                        pointArea= area(i) /REAL(integralSteps) /
& REAL(integralSteps)

                        DO 230 p= 0, integralSteps -1

                            DO 240 q= 0, integralSteps -1

                                DO 250 k= 1, 3
                                    point(k)= centroid(i,k) - sidel(i,k) /2. +
& sidel(i,k) /integralSteps /2. + REAL(p) *sidel(i,k) /
& integralSteps - side2(i,k) /2. +
& side2(i,k) /integralSteps /2. + REAL(q) *side2(i,k) /
& integralSteps

```

```

                rPoint(k)= centroid(j,k) -point(k)
250                CONTINUE

                radius= DSQRT( (rPoint(1))**2 +
&      (rPoint(2))**2 + (rPoint(3))**2 )

                influence(j,i)= influence(j,i) +
&      1. /radius /radius *
&      ( rPoint(1) *n(j,1) +rPoint(2) *n(j,2) +rPoint(3) *n(j,3) ) /
&      radius *pointArea

240                CONTINUE
230                CONTINUE

                ENDIF

                ENDIF

210                CONTINUE
200                CONTINUE

c  write(*,*) 'rhs vector:'
    DO 260 j= 1, elements
        rhs(j)= speed *COS(alpha) *n(j,1) -
&      speed *SIN(alpha) *n(j,2)
c      write(*,*) REAL(rhs(j))
260        CONTINUE

c  WRITE(*,*) 'influence matrix:'
c  DO 263 i= 1, elements
c      WRITE(*,*) ( REAL(influence(i,j)), j=1,elements )
c 263        CONTINUE

        DO 232 i= 1, elements
            rhscopy(i)= rhs(i)
            DO 234 j= 1, elements
                influenceCopy(i,j)= influence(i,j)
234            CONTINUE
232        CONTINUE

        WRITE(*,*) 'Calculating inverse of matrix.'

        CALL gaussDBLE(maxElements, elements,
&      influenceCopy, rhscopy, source)

c  write(*,*) 'Source solution:'
c  do 265 i= 1, elements
c      write(*,*) REAL(source(i))
c265        continue

C  ***** Check Gaussian elimination calculation
c  WRITE(*,*) 'Gauss Errors: These numbers should be zero!'
    DO 270 i= 1, elements
        dummy= 0.

```

```

      DO 280 j= 1, elements
        dummy= dummy + influence(i,j) *source(j)
280      CONTINUE
      IF ( (ABS( dummy- REAL(rhs(i)) ) ) .GT. 1.e-5 )
&      WRITE(*,*) '***** POSSIBLE ERROR IN GAUSSIAN ELIMINATION *****'
270      CONTINUE

C ***** Calculate added mass

      addedMass= 0.

      WRITE(*,*) 'Calculating added-mass:'

      DO 300 j= 1, bargeElements
        IF ( ( REAL(j) / REAL(NINT( REAL(bargeElements)/10. )) -
&      REAL( INT( REAL(j) / REAL(NINT( REAL(bargeElements)/10.)) ) ) )
&      .EQ. 0. ) WRITE(*,*) NINT( REAL(100 *j)/ REAL(bargeElements) ),
&      '% done...'

        phi= 0.

        DO 310 i= 1, bargeElements

          DO 320 k= 1, 3
            rij(k)= centroid(j,k) -centroid(i,k)
320          CONTINUE
            radius= SQRT( (rij(1))**2 +(rij(2))**2 +
&      (rij(3))**2 )
            IF (radius /elementSize .GE. 5. ) THEN
              phi= phi + source(i) /radius *area(i)
            ELSE
              pointArea= area(i) /REAL(integralSteps) /
&      REAL(integralSteps)

              DO 330 p= 0, integralSteps -1

                DO 340 q= 0, integralSteps -1

                  DO 350 k= 1, 3
                    point(k)= centroid(i,k) - sidel(i,k) /2. +
&      sidel(i,k) /integralSteps /2. + REAL(p) *sidel(i,k) /
&      integralSteps - side2(i,k) /2. +
&      side2(i,k) /integralSteps /2. + REAL(q) *side2(i,k) /
&      integralSteps
                    rPoint(k)= centroid(j,k) -point(k)
350          CONTINUE

                    radius= SQRT( (rPoint(1))**2 +(rPoint(2))**2 +
&      (rPoint(3))**2 )

                    phi= phi + source(i) /radius *pointArea

340          CONTINUE

```

```

330             CONTINUE

                ENDIF

310             CONTINUE
                addedMass= addedMass + phi *rhs(j) *area(j)
300             CONTINUE

                addedMass= addedMass *rho /speed /speed
                WRITE(*,*) 'Added-mass =', addedMass, ' kg,'
                WRITE(*,*) addedMass *kg2tons, ' tons.'
                bargeMass= rho * 4./3. *pi *r**3 /2.
                WRITE(*,*) 'Added-mass / barge mass =', addedMass /bargeMass
                WRITE(*,*) 'Added-mass / rho r^3 =', addedMass /rho /r**3

                STOP
                END

```

3. GaussDBLE: Matrix solution subroutine

```

SUBROUTINE GaussDBLE(limit, n, a, b, x)
C ***** GAUSSIAN ELIMINATION

INTEGER maxElements
PARAMETER (maxElements= 1000)
INTEGER limit, n
DOUBLE PRECISION a(limit, limit), b(limit), x(limit)
DOUBLE PRECISION dummy, s(maxElements), z
INTEGER i, j, k, m, p(maxElements)

C ***** Factorization phase
DO 110 i= 1, n
    p(i)= i
    s(i)= 0.d0

C ***** find scale s(i) of row i
    DO 120 j= 1, n
        IF ( DABS(a(i,j)) .GT. s(i) ) s(i)= DABS (a(i,j))
120        CONTINUE

110    CONTINUE

    DO 130 k= 1, n-1

        IF ( ( REAL(k) / REAL(NINT( REAL(n-1)/10. )) -
& REAL( INT( REAL(k) / REAL(NINT( REAL(n-1)/10.)) ) ) )
& .EQ. 0. )
& WRITE(*,*) NINT( REAL(100 *k)/ REAL(n-1) ), '% done...'

C ***** select the next pivot row
        dummy= 0.
        DO 140 i= k, n
            IF ( DABS( a(p(i), k) ) /s(p(i)) .GT. dummy ) THEN
                j= i
                dummy= DABS( a(p(i), k) ) /s(p(i))
            ENDIF
140        CONTINUE

C ***** switch p(k) <--> p(j)
        m= p(k)
        p(k)= p(j)
        p(j)= m

        DO 150 i= k+1, n
C ***** calculate and store multiplier z back in matrix
            z= a(p(i),k) /a(p(k),k)
            a(p(i),k)= z
C ***** subtract multiplier times pivot row
            DO 160 j= k+1, n
                a(p(i),j)= a(p(i),j) -z *a(p(k),j)
160            CONTINUE

```

```

150          CONTINUE

130          CONTINUE

C ***** Solution phase
C ***** subtract multiplier times pivot row R.H.S. b(p(k))
DO 210 k= 1, n-1
    DO 220 i= k+1, n
        b(p(i))= b(p(i)) -a(p(i),k) *b(p(k))
220          CONTINUE
210          CONTINUE

C ***** back-substitute to solve for x(i)
DO 230 i= n, 1, -1
    dummy= 0.
    DO 240 j= i+1, n
        dummy= dummy+ a(p(i),j) *x(j)
240          CONTINUE
    x(i)= ( b(p(i)) -dummy ) /a(p(i),i)
230          CONTINUE

RETURN
END

```

Appendix B

Sample Calculations

The following are transcripts of sample runs of the WESbargeBEM and WESSphereBEM FORTRAN programs listed in Appendix A (Sections 1 and 2).

1. JOHB

```
run wesbargedbem8
Enter barge wetted length, ft:
195
Enter barge wetted beam, ft:
35
Enter barge draft, ft:
10
Enter barge bow rake angle, degrees (90 = vertical):
45
Enter barge stern rake angle, degrees (90 = vertical):
45
Enter water depth, ft:
18
Enter barge speed, ft/s:
3
Enter barge distance from wall, ft:
70
Enter barge angle to wall, degrees:
(0 = parallel, 90 = perpendicular)
0
Total elements:          300
Calculating influence matrix:
    10% done...
    20% done...
    30% done...
    40% done...
    50% done...
    60% done...
    70% done...
    80% done...
    90% done...
    100% done...
Calculating inverse of matrix.
    10% done...
```

20% done...
 30% done...
 40% done...
 50% done...
 60% done...
 70% done...
 80% done...
 90% done...
 Calculating added-mass:
 12% done...
 24% done...
 36% done...
 48% done...
 60% done...
 72% done...
 84% done...
 96% done...
 Added-mass = 1404431. kg,
 1548.385 tons.
 Added-mass / barge mass = 0.7659771

2. SOHB

```
run wesbargedbem8
Enter barge wetted length, ft:
175
Enter barge wetted beam, ft:
26
Enter barge draft, ft:
10
Enter barge bow rake angle, degrees (90 = vertical):
45
Enter barge stern rake angle, degrees (90 = vertical):
45
Enter water depth, ft:
18
Enter barge speed, ft/s:
3
Enter barge distance from wall, ft:
70
Enter barge angle to wall, degrees:
(0 = parallel, 90 = perpendicular)
0
Total elements:          300
Calculating influence matrix:
    10% done...
    20% done...
    30% done...
    40% done...
    50% done...
    60% done...
    70% done...
    80% done...
    90% done...
    100% done...
Calculating inverse of matrix.
    10% done...
    20% done...
    30% done...
    40% done...
    50% done...
    60% done...
    70% done...
    80% done...
    90% done...
Calculating added-mass:
    12% done...
    24% done...
    36% done...
    48% done...
    60% done...
    72% done...
    84% done...
    96% done...
Added-mass =    1069359.    kg,
```

1178.968 tons.
Added-mass / barge mass = 0.8802806

3. Hemisphere

```
run wesspherebem
Enter hemisphere radius, ft:
1
Enter barge speed, ft/s:
1
Do you want to consider wall and bottom distances?
Enter 1 for yes, 0 for no:
0
Enter approximate # of elements along the draft:
13
Total elements:          918
Calculated area, actual area:
  0.5848469      0.5837271
Calculating influence matrix:
    10% done...
    20% done...
    30% done...
    40% done...
    50% done...
    60% done...
    70% done...
    80% done...
    90% done...
Calculating inverse of matrix.
    10% done...
    20% done...
    30% done...
    40% done...
    50% done...
    60% done...
    70% done...
    80% done...
    90% done...
Calculating added-mass:
    10% done...
    20% done...
    30% done...
    40% done...
    50% done...
    60% done...
    70% done...
    80% done...
    90% done...
Added-mass = 22.91457      kg,
  2.5263317E-02 tons.
Added-mass / barge mass = 0.3863743
Added-mass / rho r^3 = 0.8092206
```

Form Approved
OMB No. 0704-0188

1. REPORT DATE (DD-MM-YYYY)
August 2003

3. DATES COVERED (From - To)

Analysis of the Added Mass of a Barge in Restricted Waters; Phase 2 Model

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER	
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
10	10
11	11
12	12
13	13
14	14
15	15
16	16
17	17
18	18
19	19
20	20
21	21
22	22
23	23
24	24
25	25
26	26
27	27
28	28
29	29
30	30
31	31
32	32
33	33
34	34
35	35
36	36
37	37
38	38
39	39
40	40
41	41
42	42
43	43
44	44
45	45
46	46
47	47
48	48
49	49
50	50
51	51
52	52
53	53
54	54
55	55
56	56
57	57
58	58
59	59
60	60
61	61
62	62
63	63
64	64
65	65
66	66
67	67
68	68
69	69
70	70
71	71
72	72
73	73
74	74
75	75
76	76
77	77
78	78
79	79
80	80
81	81
82	82
83	83
84	84
85	85
86	86
87	87
88	88
89	89
90	90
91	91
92	92
93	93
94	94
95	95
96	96
97	97
98	98
99	99
100	100

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER
33143

University of Wisconsin-Platteville, Department of Mechanical Engineering, 1 University Plaza, Platteville, WI 53818-3099;
The Johns Hopkins University, Department of Civil Engineering, 3400 N. Charles Street, Baltimore, MD 21218-2686

8. PERFORMING ORGANIZATION REPORT NUMBER	
--	--

ERDC/ITL TR-03-6

U.S. Army Corps of Engineers, Washington, DC 20314-1000;
U.S. Army Engineer Research and Development Center, Information Technology
Laboratory, 3909 Halls Ferry Road, Vicksburg, MS 39180-6199

10. SPONSOR/MONITOR'S ACRONYM(S)	
----------------------------------	--

11. SPONSOR/MONITOR'S REPORT
NUMBER(S)

Approved for public release; distribution in unlimited.

14. ABSTRACT

A theoretical analysis of the added mass of barges in restricted waters is presented. The added-mass behaviors with varying waterway and barge parameters are predicted. A numerical technique is used to solve the potential-flow boundary-value problem. The analysis assumes that the Froude number is low. A restriction that the Froude number be less than or equal to 0.1 ensures that the surface waves will be small; this corresponds to a barge speed of 8 fps (2.4 m/s). For this low-speed condition, the added mass is unaffected by the barge speed. The separation between the barge and the wall is seen to have influence near the wall.

Added mass
Green's function

Numerical solution
Potential flow

Wall effect

a. REPORT

UNCLASSIFIED

b. ABSTRACT

UNCLASSIFIED

c. THIS PAGE

UNCLASSIFIED

17. LIMITATION OF ABSTRACT

18. NUMBER OF PAGES

19a. NAME OF RESPONSIBLE PERSON

19b. TELEPHONE NUMBER (include area code)

49